A Cluster-Ring Topology for Reliable Multicasting

Lenka Motyčková
ITN, Linköpings University
S-601 74 Norrköping, Sweden
lenmo@itn.liu.se

David A. Carr
IDA, Linköpings University
S-581 83 Linköping, Sweden
davca@ida.liu.se

Esther Jennings
Dept. of Computer Science
California State Polytechnic University
Pomona, CA 91768, USA
ehjennings@csupomona.edu

Abstract *Applications based on multicasting such as real-time simulations or shared editors require that all data packets are delivered safely in a reasonably short time. Trying to assure such a high quality of service centrally, can easily overload both the network and the source. One technique to prevent this is clustering (organizing multicast group members into subgroups). We present an algorithm to make clustering as natural as possible by building clusters from groups of nodes that are close together in dense parts of the network. The cluster building algorithm uses only local knowledge and executes in parallel for all nodes. We have simulated our algorithm and find that it builds reasonable clusters for the topologies tested. Finally, we propose an extension of RMP, a token-ring-based, reliable multicast protocol, using our algorithm to build a ring of tree-organized clusters. This combination makes the resulting protocol scalable, which the original RMP was not.*

Keywords: clustering, reliable multicasting, scalability, token ring

1 Introduction

Many multimedia applications based on multicasting require that all data packets are delivered in a reasonably short time. As the basic multicast protocols are just best effort protocols, receivers must have the possibility to request a missing data packet and have it delivered so that a sound or picture can be presented correctly. The reliable enhancement of a protocol creates additional packets that must be transferred between receivers and sources. This means that:

1. More network *bandwidth* is consumed by a reliable protocol. So, we need an efficient algorithm that reduces the amount of control and retransmission traffic while still keeping multicasting reliable.

2. The *delay* before a missing data packet is delivered should be minimized. Some applications require fast retransmissions, as receiving a missing packet after a substantial delay is useless for receivers.

3. With the growing size of the Internet as well as growing interest in multicast applications, one can expect large multicast groups and groups that are widely spread. Because routers in a multicast group cooperate when data packets are acknowledged and retransmitted, growth of multicast groups generally increases the *load* on all intermediate routers. As this can easily overload a router, arrangements must be made to keep network load independent of the number of receivers or members in a group.

Protocols that meet this requirement are called *scalable*. We propose a protocol that uses *clustering* to achieve scalability. Clustering is a known technique in the area of distributed network computing. The aim is to use local properties of the network to speed up computation (by sharing information, preferably inside of a local group or cluster) and to decrease overall load on the network by performing as much computation as possible locally and sending globally only data that (in some sense) represents all nodes in a cluster.

A constant load on all members of a multicast group, is one of the basic issues in multicast protocol design. What limitations must be imposed on the network topology, to have constant load? As routers are able to handle just a limited number of receivers, the degree of a node in a network graph is bounded by a constant. Keeping the number of messages sent over each link constant is the first issue. The second issue is to keep the number of clusters constant, so that the number of nodes communicating with the source–the cluster leaders–is constant. These limitations ensure that the throughput of all nodes in a multicast group is independent of the size of the multicast group. For a more detailed load explanation see the protocol description in Section 3.

In reliable multicast protocols, a packet's loss is detected by receivers and a negative acknowledgment (NAK) is used to inform the source of packet loss. A positive acknowledgment (ACK) is used to inform the source that a packet is successfully received by the receiver(s). If an ACK/NAK is sent directly to the source from every receiver for a large group of receivers, this may cause ACK-/NAK-implosion. The source spends nearly all of its time processing control messages. On the other hand, sending NAKs for only the missed packets is not sufficient to insure reliability [9]. Also, nodes that are incident to congested links, might repeat NAKs and overload neighboring nodes. Avoiding ACK/NAK-implosion is important property of a scalable protocol.

Our proposed protocol avoids implosion by using a clustered structure where ACK/NAKs are processed locally within each cluster whenever possible. The source receives just a limited number of acknowledgements.

1.1 Comparison to Previous Work

Most of the reliable multicast protocols proposed to date, divide receivers into groups, to obtain a structure suitable for acknowledgement processing. The tree-based, reliable, multicast protocol RMTP uses clustering [11]. However, its clusters have only depth one. Also, the *designated routers* (cluster leaders) in RMTP are chosen statically for the multicast session. The clique clustering of [8] uses a greedy approach to create local groups of cliques. Packets are routed over the shortest paths between boundary nodes in cliques. Cliques that exist in the actual Internet topology are of small degree (typically 2 or 3), so this kind of cluster represents a very small fraction of the network. In the (rooted) shared ACK-tree of [10], a node forms a local group with B of its children, where B is a parameter. A child is any node within a distance less than a predefined delay. The tree structure can be considered as a clustering where a cluster is formed by children of a single node, which becomes a cluster leader responsible for its successors.

A known principle for organizing acknowledgment of multicast messages is a circulating token principle. It guarantees totally-ordered message delivery, naming service, good resilience and reasonable performance. First described in [2] for reliable and ordered broadcast, it was later used as a basis for the reliable multicast protocol RMP [13]. The protocol can handle concurrent multicast sessions in a multicast group, so the topology structure used for acknowledgements is a shared one. The basic entity here is a token site that communicates with all receivers and all sources. It represents all receivers for a source by sending only one positive acknowledgement per message. The source retransmits periodically until it receives an ACK from the token site. For receivers, the token site represents a source in the sense that NAKs are sent to the token site instead of directly to the source. The third kind of interaction takes place between token sites. A higher number of token sites is necessary as a single token site may crash and impact reliability. Also, the NAKing mechanism does not guarantee retransmission in the case of lost messages. Interaction between two subsequent token sites is based on an ACKing principle. The next token site does not accept a token unless it has received all messages sent so far. This mechanism ensures that each token site receives all multicast messages within the delay proportional to the time needed by a token to complete one round through all token sites. If each message is stored at L token sites simultaneously, then the protocol is L-resilient. This is achieved by keeping a message at the last L token sites and only erasing it as the token advances.

1.2 Our Aim

The main contribution of this paper is twofold. The first is to propose a cluster decomposition of a multicast group which is suitable for acknowledging data packets and for local retransmissions in reliable multicast. Second, we propose an acknowledgement mechanism for reliable multicast, that is based on a token circulation over the set of clusters. It combines features of token ring and tree-based protocols.

The key idea behind our proposed structure is to divide a multicast group graph into clusters that correspond to local groups of receivers. Our structure is a *disjoint cluster graph* where the clusters are densely connected internally, and inter-cluster links are sparse. This structure has the following desirable properties:

Topology preserving clustering: Local groups are formed in a manner that truly reflects the actual connectivity of the multicast group, rather than forming them arbitrarily. They are built preferably around nodes with a high degree (large number of neighbors).

Optimal delays within clusters: Clusters in the dense parts of a network have many members, but the height of their intracluster spanning tree is low compared to the number of cluster nodes.

Scalability of reliable multicast protocol: The state information kept at each receiver is independent of the number of receivers in the multicast group. To achieve this, the number of token sites and the number of successors in tree subnetworks must be bounded by a constant.

2 Cluster-Ring Structure for Acknowledgments

2.1 Assumptions

Using a multicast routing tree(s) provided by best-effort multicast routing protocols such as DVMRP [4], CBT [1], OCBT [12], or PIM [5], we organize the receivers of a multicast group into a shared ACK-cluster structure.

Beginning with an underlying shared multicast tree, we define our graph G as the connected subgraph of the network induced by the vertices of the given multicast routing tree(s). G contains all the vertices of the multicast routing tree(s), plus all other IP-network connections induced by these vertices. We only consider router nodes in our graph G since receivers are nodes that are directly connected to routers (as peripheral subtrees).

The aim is to create clusters which are used as an ACK-structure. The clusters should capture the entire local dense mesh of receivers. If an area is densely populated with receivers and there is a reasonable branching from each node, the diameter of the cluster is logarithmic in terms of the number of cluster nodes. Nodes in a cluster are close to each other (small delay), and there are many near the leader.

2.2 Clustering Algorithm

The problem that we consider here is to detect dense subgraphs in a network graph, and then cover the graph by disjoint clusters, built around the dense subgraphs. As distributed computation is natural for networking, we propose a parallel solution to building clusters, Let us assume a maximal value of the cluster diameter D, which defines the delay between the leader and the farthest cluster member and $k = Deg_{est}$, an estimate of the highest degree of any node in the network graph.

1. Every node checks its degree. If its degree is at least k, it starts to build a cluster. Note this occurs concurrently, so clusters compete for members.
   - Each potential leader sends an "offer" message to its neighbors.
   - A node without a leader waits a short delay after receiving the first "offer". If more than one "offer" is received, it accepts the "offer" of the node with the lowest network address and replies with an "accept" message.
   - If a leader node receives an offer from a lower address node, it accepts and notifies its cluster members that it is no longer a leader. It will also reject any further accept messages.
   - A node with degree at least k will forward an offer to its neighbors as long as it is less than D from the offering node.
2. To cover sparser areas, repeat step 1 for $k = k - 1$, until the $k = 3$. Nodes with leaders determined for higher k do not change leaders; however, they do forward offer messages as above.
3. Orphans: nodes, not included in any cluster and having only one adjacent edge, attach to the closest cluster.
4. Chains connecting clusters are divided among closest clusters.
5. Check the number of clusters: If too many $->$ merge small clusters into large, then decrease the starting k.

Figure 1: Algorithm description

that performs better in distributed environments, even if conflicts occur when clusters clash.

A dense subgraph can be detected by deleting vertices with a small degree until the one with the required dense neighborhood is found. If an empty set is found, the procedure is repeated with different density criteria. A second possibility [7] (more suitable if a large subgraph is required) is to look for the vertices with the largest degree $deg(v)$. We apply the second strategy in our algorithm.

After detection of a dense kernel, a cluster is created as a neighborhood of v by growing the kernel according to an expansion condition. We are looking for subgraphs including a substantial fraction of the edges with respect to the number of edges in the entire graph. The authors of approximation algorithms for the related problem of computing the densest l-vertex subgraph of a given graph [7] suggest an expansion condition that requires the number of edges touching the subset is at least $|E|/2l * |V'|$, where E is a number of edges in G and V' is number of nodes in a subset. Their condition is based on observation that for large l there always exists a l-vertex subgraph that contains a substantial portion of edges in the entire graph.

Our aim is not to detect the single densest subgraph but to detect a number of dense subgraphs. Our condition is weaker compared to [7]: The number of edges touching the subset is $Deg_{est} * |V'|$, where Deg_{est} is an estimate of a maximal degree in a graph. Kernels must be enlarged only with nodes that increase the number of edges in a subgraph by a fraction proportional to the density of the whole graph. After detection of all dense kernels we start growing clusters around them based on our expansion condition. The condition is based on the estimate of the graph density, and we grow a subgraph as long as the condition is fulfilled until the limit for a maximum diameter of a subgraph is achieved. If the graph is not exhausted yet, we relax the expansion condition and repeat the procedure in the remaining graph. The aim is to cover the whole graph by clusters, but there might be parts of the network, where the density requirement cannot be fulfilled: topologies with long peripheral chains of receivers. In this case we attach remaining nodes to the closest clusters. Even if such a topology is inefficient, there is no better solution.

As the number of clusters must be limited by a constant in order to keep a constant size token ring, we check this parameter after each iteration. If the bound is exceeded, it is necessary to increase the size of clusters. This is achieved by decreasing the density estimate parameter k in the next iteration.

3 Simulation

In order to get an idea of how our clustering algorithm performed we simulated it. We generated several 60-node networks using the Tiers Network Topology Generator [3],[6]. The networks had a 10-node backbone, with five, 5-node, medium-area networks attached to the backbone, and five single-node LANs attached to each MAN. All redundancies were set to 1, except for the intra WAN and MAN to WAN which were set to 2. An example clustering can be seen in Figure 2. The simulation assumed: all links had equal delays, no messages were lost, all nodes had equal response times. These assumptions mean that ideal performance will be achieved. Either 5 or 6 clusters were generated for each network topology. All topologies generated small clusters of three or four nodes and at least one large cluster. In one case, the large cluster contained over half of the nodes.

4 The Reliable Multicast Protocol

In this paper, we propose a reliable protocol that combines a token-based ACK mechanism with a tree-based one. Clusters are logically

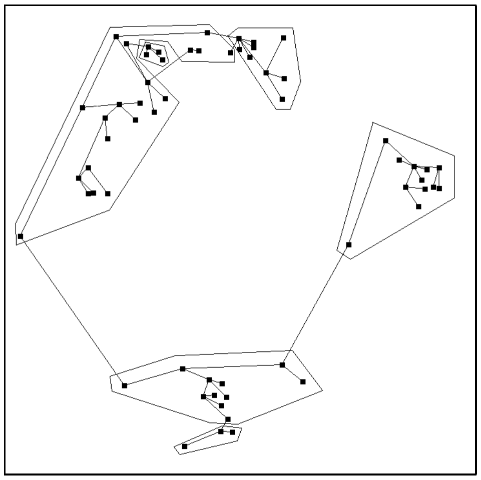

Figure 2: Example of clusters produced by our algorithm

connected to a token ring; each cluster is represented by a leader, that acts as a token site. Among cluster leaders the RMP-like protocol is used. Inside of clusters, we apply a tree-based protocol, e.g. RMTP [11]. ACK packets are aggregated from the leaves to the root inside of a cluster on a local level, and communicated to sources by token sites on a global level. ACKs are multicast by cluster leaders (token sites), according to the basic token ring protocol [13]. Cluster leaders ask for retransmissions at the current token holder. Composing an ACK-token-ring structure of clusters instead of single nodes, leads to more parallelism in gathering ACKs. For the details of ring reconfiguration, resiliency or congestion control, refer to the basic protocols [13] and for tree-based routing of ACK/NAK packets refer to [11].

4.1 Scalability of Token-Ring Protocols

The Reliable Multicast Protocol (RMP) uses all receivers as token sites, i.e. all group members are organized in a logical ring. RMP distributes the communication load between all sites. Experimental results presented in [13] show that the performance stays roughly constant independent of the number of receivers. RMP's authors also suggest including a server into the ring of multicast group members. The server then communicates multicast packets to/from non-member clients. These features move the protocol towards scalability, but in fact do not guarantee a constant load, independent of multicast group size, at each processor engaged in a multicast session. Managing a logical ring of the size of the group and sending the ordered list of all members constitutes a load that grows proportionately with the number of receivers. The need to include all receivers comes from the insufficiency of NAKing mechanisms.

We propose to introduce the following limitations in order to guarantee scalability of an RMP-like protocol:

- For the token ring: bound the number of token sites. The bounding constant is based on the capacity of the token sites to process control messages of the protocol that maintain the ring structure.

- For clusters: Each cluster member keeps the list of its successors, the number of which is bounded by a constant, and it knows its leader. A cluster leader receives messages only from its successors.

4.2 Cluster-Ring Topology

By making the idea of a server more general, we get a protocol that is provably scalable. To make the protocol scalable, the size of the ring has to be constant. In practice, it means that only a constant fraction of receivers can serve as token sites. In order to guarantee scalable reliable multicast for the whole group, members that are not attached directly to the ring, are connected indirectly, through their cluster leader that serves as a token site. Cluster members are attached to an intracluster spanning tree, where the leader is the root and communication is on the parent-child principle. Scalability of this type of protocol has been proven in [10].

The structure that we get is a ring of clusters, where for each receiver, the reliable and ordered multicast is guaranteed by the ring protocol. In order to provide the same service for the cluster members, the cluster leader (the receiver on a ring) is responsible for reliable delivery to members. Note that, primary multicast is performed by the unreliable IP protocol, and cluster leaders provide only retransmissions in the case of packet loss. This implies that a leader cannot erase packets from its memory until it is ACKed by all cluster members. To keep the protocol L-resilient, the token site does not accept the token until all packets sent so far are acknowledged by its cluster members. Members ask for retransmissions at their parents and then ACKs confirm receipt of a packet by all members, so that the leader can delete a data packet from its memory. For details of tree-based protocols refer to [11].

Acknowledgements sent by token-sites are multicast - this means that all receivers including those that are just cluster members and not connected to a token ring, receive ACK packets. This also guarantees total ordering inside a cluster. The positive feature is that our topology supports optimal delays when communicating ACKs and retransmitting inside of intracluster trees. Clusters are created in dense subnetworks, so election of token sites is therefore not done randomly, but it is based on the actual topology of the multicast group.

5 Conclusion

We present an algorithm for building clusters that follows the topology of multicast groups. The method selects high-degree nodes, attaches their neighbors, and expands around high-degree neighbors. The algorithm insures reasonably fast response by bounding the radius of a cluster. We simulated our algorithm and found reasonable clustering for the topologies investigated. We further show how this algorithm can be used to achieve scalability for the RMP reliable multicast protocol. This improvement creates a token ring of clusters where a tree-based protocol is used within clusters.

6 Acknowledgments

This research was supported by the Linköping University CENIIT program, projects 99.9 and 00.4. We would also like to thank Carl Rollo for his help in proofreading this paper.

References

[1] Ballardie, T., Francis, P., Crowcroft, J. Core based trees (CBT): An architecture for scalable inter-domain multicast routing. *Proc. ACM SIGCOMM* (1993), 85–95.

[2] Chang, J. M., Maxemchuk N., F. Reliable broadcast protocols. *ACM Trans. on Comp. Systems*, 2(3), 251–273, Aug 1984.

[3] Calvert,K. L., Doar, M. B., Zegura, E. W. Modeling internet topology. *IEEE Communications Magazine*, 35(6), 160–163, June 1997.

[4] Deering, S., Cheriton, D. Multicast routing in datagram inter-networks and extended lans. *ACM Trans. on Comp. Systems*, 8(2), 85–110, May 1990.

[5] Deering, S., Estrin, D., Farinacci, D., Jacobson, V., et al. An architecture for wide-area multicast routing. *Proc. ACM SIGCOMM*, (1994) 126–135.

[6] Doar, M. B. A better model for generating test networks. *IEEE Global Telecommunications Conference/GLOBECOM'96*, London, November 1996, 86–93.

[7] Kortsarz, G., Peleg, D. On choosing a dense subgraph. *Proc. 34-th FOCS* (1993), 692–701.

[8] Krishna, P., Vaidya, N., Chatterjee, M., Pradhan, D. A cluster-based approach for routing in dynamic networks. *ACM SIGCOMM Computer Communication Review*, 27(3), 49–64, Apr 1997.

[9] Levine, B., Garcia-Luna-Aceves, J.J. A comparison of known classes of reliable multicast protocols. *Proceedings of International Conference on Network Protocols (ICNP-96)*, 112–121, Columbus, Ohio, Oct 29–Nov 1, 1996.

[10] Levine, B., Lavo, D., Garcia-Luna-Aceves, J.J. The case for reliable concurrent multicasting using shared act trees. *Proc. of ACM Multimedia*, 365–376, Boston, MA, USA, Nov 1996.

[11] Lin, J. C., Paul, S. RMTP: a reliable multicast transport protocol. *Proc. INFOCOMM'96*, 1414–1424, Mar 1996.

[12] Shields, C. *Ordered core based trees.* Master's thesis, University of California – Santa Cruz (1996).

receiver-

[13] Whetten, B., Montgomery, T., Kaplan, S. A high performance totally ordered multicast protocol. *Proc. Theory and Practice in Distributed System*, LNCS vol. 938, Sep 1995.